First Hit Fwd Refs

End of Result Set

☐ Generate Collection | Print

L9: Entry 1 of 1 File: USPT Feb 1, 1994

DOCUMENT-IDENTIFIER: US 5283838 A
TITLE: Neural network apparatus

Application Filing Date (1):
19910415

Detailed Description Text (114):
In the modified algorithm, when the learning vectors initially presented in self-organization learning are wrong learning vectors (for example, learning vectors based on the Kanji character " (Ki)" (meaning "a tree") in self-organization learning for the category " (Dai)"), extremely peculiar learning vectors, noisy learning vectors, etc., the weight vector which exhibits the greatest degree of similarity is not always changed or updated by the correct learning vectors which are input later. This means that the wrong learning is not corrected. Accordingly, in the learning process in the neural network according to this embodiment, it is required that wrong, extremely peculiar, or noisy learning vectors are not presented initially in the self-organization learning.

Detailed Description Text (125):
In each of the above-described embodiments, learning is conducted using a Kohonen type neural network. The invention is not restricted to this, and alternatively, can be applied to a perceptron type neural network. In this case, the output nodes Lkm shown in FIG. 1 may be used as intermediate layer nodes of the perceptron type neural network, and the input node 2 shown in FIG. 1 as an input layer node of the perceptron type neural network. Namely, a plurality of the intermediate layer nodes of the perceptron type neural network are allocated to each of categories of objects to be identified. Then, self-organization learning is performed for the intermediate layer nodes and the input layer node, and weights of the intermediate layer nodes are initialized. Next, teacher-supervised learning such as back-propagation is carried out for the entire neural network in which the weights of the interlayer nodes have been initialized.

First Hit Fwd Refs

Generate Collection

L11: Entry 11 of 11 File: USPT Oct 22, 1996

DOCUMENT-IDENTIFIER: US 5568181 A
TITLE: Multimedia distribution over wide area networks

Application Filing Date (1):
19950501

Brief Summary Text (6):
In accordance with the illustrative embodiment of the present invention, the distribution of video files over large geographical areas makes use of local video caches along with efficient distribution of such video files to the local caches. High speed local area networks are then able to deliver the video files locally from the local cache in real time, while a slower wide area network is able to transfer video files from one, or more centralized video storage libraries to the local caches at the slower, non-real time rates common to such wide area networks. More particularly, user access to video files utilizes one of three different algorithms, depending on the request and the local availability of a file. If the file is already available locally, for example, the user may obtain full access (browsing, playback, rewind and multiple viewing) over local area network facilities such as those currently available today. If a request specifies a future time for access, a remote file can be scheduled for transfer to the local cache at any convenient time or times prior to viewing. Finally, if a request specifies a video file which is not in the local cache, a "preface" of the video file is immediately transferred to the local cache. The preface is a predetermined initial portion of the video file having a playback duration just long enough to balance the time required to transfer the remainder of the video file to the local cache with the time to play back the entire video file. This latter type of file access is called "speed match" playback.

Detailed Description Text (4):
The video distribution management portion of FIG. 1, providing file service functions, is comprised of the video library 11, the wide area server 10, WAN 13 and video distribution management system 12. The video playback portion, providing interactive video access, comprises local area server 14, LAN 16, local area video cache 15 and a plurality of video display stations like station 17. Local area server 14 serves as a rate changing interface between the high speed LAN 16 and the lower speed WAN 13. As will be described hereinafter, video distribution management system (VDMS) 12 receives requests from all of the video display stations, such as station 17, connected to all of the LANs, such as LAN 16, connected to WAN 13, and provides interactive playback of video files by downloading such video files from library 11 to local caches 15 at the transmission rate of WAN 13, and then provides interactive, real time video playback of these same files from local cache 15 to stations 17 over LAN 16.

Detailed Description Text (24):
It can be seen that the processes described in FIGS. 2 through 5 cooperate to provide efficient and economical distribution of video files from a remote video library to a large number of widely distributed video file users, using a wide area network and a plurality of local area networks as the transmission vehicles for such video distribution.

First Hit Fwd Refs

Generate Collection

L17: Entry 6 of 9 File: USPT Jul 7, 1998

DOCUMENT-IDENTIFIER: US 5778393 A
** See image for Certificate of Correction **
TITLE: Adaptive multitasking for dataset storage

Application Filing Date (1):
19960520

Brief Summary Text (15):
The invention affords a number of distinct advantages. For example, the invention minimizes the processor overhead expended in activating data storage devices since data storage devices are only invoked as needed. Furthermore, the invention reduces the waiting time to store data by dividing excessively large blocks of data and distributing their storage among the data storage devices. The invention is therefore useful, for example, in a recovery system of a data storage system, where data is efficiently stored in a journal volume for later use in recovering a failed primary storage volume.

L10: Entry 34 of 36 File: USPT Jan 5, 1999

DOCUMENT-IDENTIFIER: US 5857072 A
TITLE: System and method for distributing data simultaneously to multiple computers on a network, with advanced notice to intended recipients

Application Filing Date (1):
19960430

Brief Summary Text (9):
Connectionless operations manage user PDUs as independent and separate entities. No relationship is maintained between successive data transfers, and minimal records are maintained concerning the ongoing communications process through the network. In contrast to connection-oriented service, connectionless service provides neither positive acknowledgments nor negative acknowledgments regarding the data transmission. Thus, by its very nature, connectionless service can achieve significant independence from: (a) specific protocols within a subnetwork, (b) subnetworks from each other, and (c) subnetworks from user-specific protocols. Additionally, connectionless networks are not concerned with flow control or any type of resequencing operations at the final destination. Connectionless networks may also allow multiple copies of the same message to arrive via bridges. As noted, each PDU is handled as an independent entity such that data units can take different routes to avoid failed nodes or congestion at a point in the network. However, connectionless protocols do consume more overhead than their connection-oriented counterparts in relation to the length of the headers and in proportion to the amount of user data in the PDU.

Brief Summary Text (11):
A significant limitation of each of the above-referenced file distribution packages relates to the method in which data is distributed. Specifically, in each of these distribution packages, all data is distributed to multiple client computers one at a time. Thus, data sent to one hundred client computers, for example, must be sent one hundred times (once to each computer) and thus takes one hundred times longer than would be required to send the data to one client computer. Consequently, when large numbers of client computers associated with the network require a data distribution, existing file distribution systems require a significant amount of network bandwidth and increased distribution time.

Detailed Description Paragraph Table (1):
Structure for message to ask client for information about its current data files: typedef struct tagSTRUCT.sub.-- WHAT.sub.-- U.sub.-- GOT { CMNG.sub.-- HEADER xHeader; // ucMessageID == MSG.sub.-- WHAT.sub.-- U.sub.-- GOT WORD wNumOfGroups; // max MAX.sub.-- NUM.sub.-- GROUPS WORD wNumOfFileNames; // max MAX.sub.-- NUM.sub.-- DIR.sub.-- ENTRIES GROUP.sub.-- INFO xGroup[MAX.sub.-- NUM.sub.-- GROUPS]; FILE.sub.-- NAME xFileName[MAX.sub.-- NUM.sub.-- DIR.sub.-- ENTRIES]; } STRUCT.sub.-- WHAT.sub.-- U.sub.-- GOT; Structure for message to initiate a file transfer: typedef struct tagSTRUCT.sub.-- FILE.sub.-- HEADER { CMNG.sub.-- HEADER xHeader; // ucMessageID == MSG.sub.-- FILE.sub.-- HEADER WORD wNumOfGroups; // max MAX.sub.-- NUM.sub.-- GROUPS WORD wNumofRecords; // change to 32-bit if needed GROUP.sub.-- INFO xGroup[MAX.sub.-- NUM.sub.-- GROUPS]; FILE.sub.-- NAME xFileName; FILE.sub.-- INFO xFileInfo; DWORD dwActivationTime; } STRUCT.sub.-- FILE HEADER; Structure for message to send

partial data for a data transfer: typedef struct tagSTRUCT.sub.-- FILE.sub.-- RECORD { CMNG.sub.-- HEADER xHeader; // ucMessageID == MSG.sub.-- FILE.sub.-- RECORD long int lRecNum; // -1 to indicate download aborted FILE.sub.-- DATA xFileData; } STRUCT.sub.-- FILE.sub.-- RECORD;

First Hit Fwd Refs

☐ Generate Collection

L10: Entry 33 of 36 File: USPT May 11, 1999

DOCUMENT-IDENTIFIER: US 5903566 A
TITLE: Method for distributing program code to intelligent nodes in a wireless mesh data communication network

Abstract Text (1):
A large data file is distributed to a number of nodes in a data communication network by a process of distributed downloading. Destination nodes are informed of the location in the network of the large data file and are directed to receive the large data file by requesting that blocks of data containing the file be transmitted to them from the designated source node. The destination nodes control the file transfer. The large data file may contain program code for updating network software.

Application Filing Date (1):
19940624

Brief Summary Text (3):
Packet communication is a form of data communication whereby segments or packets of data are routed with error checking and confirmation of receipt. Packets may be transmitted directly between a source node and a destination node or may be relayed via a number of relay nodes. Several methods of data packet routing are known.

Brief Summary Text (4):
Some methods of packet communication are directory-based routing and non-directory-based routing. According to directory-based routing method, the address in the header of a packet is used as an index to a directory of packet routing lists stored in a source node. The source node must maintain and continuously update a routing list for each node in the network. In non-directory-based routing, the complexities associated with directory-based routing techniques are avoided. There is no need to store connectivity information for each transmitting node in the network, thus reducing the amount of overhead processing that must be done by the network to preserve network connections. However, non-directory-based routing techniques generally do not permit network parameter optimization. A number of patents have issued to the assignees of the present invention concerning various aspects of data network operation including U.S. Pat. Nos. 5,079,768; 5,115,433; and 5,007,052.

Brief Summary Text (5):
A common task that must be periodically performed in any network is the updating of network software in each of the physically separate machines that make up the network. The network software resides individually in each physically separate network node and controls how that machine interacts with the network. In general, changing the network software involves installing the new software on all of the machines in the network. This may be done manually by a human operator at each machine but is preferably done over the network in such a way that one node on the network directs the other nodes to accept data transmissions that include the new program code and then directs the nodes to execute the new code.

Brief Summary Text (6):
Typically, the data file that contains the new executable program code is very

large when compared to other files transmitted on the network and may take many minutes to several hours to transmit, even just between two nodes. Updating all the nodes on a network can therefore consume significant network resources, particularly if the program code file is routed independently from the source node to each node in the network.

Brief Summary Text (7):
Two methods known in the art for program downloading are direct transmission and code float.

Brief Summary Text (9):
This method has several drawbacks in practice, especially when utilized in attempting code download to a multitude of nodes in a geographically distributed broadcastless communication environment. First, when a large number of nodes require simultaneous updating, keeping track of the progress of the individual downloads becomes logistically difficult. In addition, the described method requires that the medium over which the download is proceeding be error free. To create an error free connection over any of several unreliable media requires substantial computing resources to be dedicated to the maintenance of each connection. In practice, this limits the number of simultaneous connections (downloads) that can exist. Finally, the path between the code source and the destination node must remain intact throughout the download. This requirement places possibly severe demands on the stability of the network topology; in environments where the network topology is changing rapidly, it may not be possible to maintain a connection for the length of time required for the complete code transfer.

Brief Summary Text (10):
Another method for the transport of executable code is often called the "code float" method. In this method, data packets containing the executable code are "floated" over the network medium to all the nodes in the network. Each node keeps track of the blocks it has received (or those it has not received and thus requires). When it has received and validated all the blocks of code it requires, the node generally sends in a report to the source stating that "it is done downloading." When all the nodes have reported in, the source can stop and free the consumed data bandwidth for other uses.

Brief Summary Text (11):
This method also has several drawbacks in practice that place limits on its effectiveness for download to geographically distinct nodes. First, the method requires that a broadcast medium (or a close approximation) be available for "floating" the code out. In addition, much data is sent out redundantly; the source does not know which blocks are still required by the receiving nodes and thus usually just keeps sending blocks in sequence until all the nodes have reported in as complete (or conversely, have reported in that they still need blocks of code). Finally, code floating by nature is best utilized when all the nodes in a network are homogeneous. It is unsuited for the download of different code bodies, for example in the case where nodes with completely different functionality are required.

Brief Summary Text (12):
What is needed is a method for distributing large files possibly containing executable code to a large number of nodes in a network that does not have the inefficiencies associated with direct transmission or code float.

Brief Summary Text (14):
According to the invention, in a wireless mesh packet communications system wherein a multitude of nodes exist that are not able to cheaply broadcast, executable code download may be accomplished by informing each destination node of ~~the~~ location (also called an address) of a source node and a description of the data file blocks

at that source that make up an executable code file. The destination (or code recipient) is completely responsible for executing and completing the code download; the source of the executable code need only respond to requests for a given block of data--no other state information or connection resources are required.

Brief Summary Text (15):
In the selected environment, this method has several critical advantages over competing approaches. First, any number of destination nodes may download code from a selected source node at a time (subject only to the maximum data bandwidth supportable by the source node). This flexibility is a benefit in operational networks. In addition, since the destination node is responsible for requesting the blocks it requires, only those blocks of data that are required will be transported through the network. This leads to an optimally efficient transfer. Finally, multiple sources may support download simultaneously. Thus, several dissimilar nodes may be updated without having to "wait their turn".

Drawing Description Text (3):
FIG. 2 is a flow chart describing program code downloading by a destination node.

Drawing Description Text (4):
FIG. 3 is a flow chart describing program code downloading by a source node responding to requests for block transfer from a destination node.

Detailed Description Text (2):
FIG. 1 shows a data network of a type in which the method according to the present invention may be employed. The network consists of nodes 10 labelled O through Z interconnected by paths 12 representing allowable communication links between nodes.

Detailed Description Text (3):
According to the present invention, executable code downloading is initiated from one initiating source node, such as node V. Node V acquires the executable code either manually or otherwise and then it is available to other nodes, such as nodes P and Y, for downloading. Once node V has acquired the executable code, a network controller sends a message to destination nodes P and Y to which node V is connected, instructing those destination nodes to request program code transmission from node V. The network controller may be one of the nodes in the network having network control capabilities or it may be a human operator. Nodes P and Y then each independently begin requesting blocks of executable code to be transmitted from node V. Nodes P and Y continue requesting blocks from node V, until they have received the full executable code file. When nodes P and Y have successfully received the full executable code file, they each inform the network controller that downloading is complete.

Detailed Description Text (4):
According to one embodiment of the invention, this distributed downloading occurs iteratively such that, once the network controller is informed by node Y that it has received the executable code file, the network controller transmits a packet to each of nodes W, Z, U, and T, instructing those nodes to request executable code transmission from node Y. Nodes W, Z, U, and T then each independently begin requesting blocks of executable code to be transmitted from node Y and continue requesting blocks from node Y, until they have received the full executable code file. Each node sends an acknowledge packet to inform the network controller that downloading is complete. Likewise, the network controller transmits a packet to each of nodes S, O, and R, instructing those nodes to request program code download from node P.

Detailed Description Text (5):
FIG. 2 depicts a flow chart illustrating the operation of a destination node

according to one embodiment of the present invention. While the steps of this process are shown in a particular order, it will be obvious to one of ordinary skill in the art that the order of some of the subprocesses contained in this process could be changed and that some steps of this process could be handled in parallel.

Detailed Description Text (6):
The process begins when a destination node receives a command packet from the network controller to download a new program code file from a particular source node (Step S1). This program file will typically have been transferred to the initial source node manually, such as by loading a floppy disk or other removable media onto the source node, or the file may be transferred to the source node over a dial-up phone line connection to the source node. The command packet contains the information the destination node needs to download the full program code file including the address of the source node, identifiers for the data blocks in the file, and an address of a node to which the destination node must send an acknowledgement when downloading is complete. The destination node then begin requesting blocks from the source node (Step S2). Once a request for a block has been made, the destination node waits to receive a block back from the source node (Step S3). When a block is received, the destination node checks to see if it requires the block for its executable code (Step S4). If the block is required, the destination node stores the block (Step S5) and checks to see if more blocks are required (Step S7). If the block is not required, the destination node drops or discards the block (Step S6) and checks to see if more blocks are required (Step S7). If more blocks are required, the destination node requests the next block (Step S2) and the process loops. If more blocks are not required, the destination node has received all the blocks that make up the executable code, it sends a message to the network controller informing it of that, and the process ends.

Detailed Description Text (7):
FIG. 3 depicts a flow chart illustrating the operation of a source node according to one embodiment of the present invention. The process begins when the source node receives a request packet from a destination node to transmit a block of data (Step T1). The source node checks to see if the request is for a valid block in its executable code (Step T2). If the block is not valid, the source node frees its packet transmit buffer (Step T4) and the process ends. If the block is valid, the source node first sends the block to the destination node (Step T3) then frees its packet transmit buffer (Step T4) and the process ends.

Detailed Description Text (8):
It will be apparent that this method of program code downloading has a number of advantages over code-float or direct transmission. First is that downloading to all nodes in a network will be faster than with direct transmission, in part because the iterative downloading may occur in parallel. Once node Y and P in FIG. 1 have each received a complete program code file, the nodes that get their program code from Y and P can do so independently and simultaneously.

Detailed Description Text (9):
This method is also especially suited for networks made up of different types of network nodes, such as stationary nodes and roaming nodes, where the different nodes require different program code. With this method, downloading of program code to the different types of nodes can occur completely independently.

Detailed Description Text (10):
A number of modifications to this method will be obvious to one of ordinary skill in the art. For example, the functions of the network controller could be transferred to each node such that once a destination node had completed program downloading, that destination node would send command packets to other nodes with which it could communicate instructing those other nodes to request program code downloading.

CLAIMS:

1. In a wireless mesh packet communications network, said network comprising a plurality of intelligent nodes interconnected by wireless links, a method for updating program code of the network, comprising the steps of:

acquiring a program code file at a source node;

dividing said program code file into blocks of a size easily transmitted on the network;

transmitting a command packet from a network controller to a plurality of destination nodes said command packet directing said destination nodes to request program code downloading from said source node; and

in response to said command packet, beginning independent program code downloading at each of said destination nodes by said destination nodes sending individual block requests to said source node for said source node to transmit blocks containing said program code.

2. The method according to claim 1 wherein said command packet includes an address for the source node and a block list identifying the blocks that comprise the program code file.

3. The method according to claim 1 wherein said command packet includes a second list of nodes referred to as secondary destination nodes and wherein said destination node sends a secondary command packet to said secondary destination nodes after said destination node has completed program code downloading said secondary command packet directing said secondary destination nodes to request blocks of said program code to be downloaded from said destination node.

4. The method according to claim 1 wherein said command packet includes an address for a network controller to which said destination nodes will send a download complete packet when said destination nodes have successfully completed program code downloading.

5. The method according to claim 3 further comprising:

when downloading is complete at a destination node, determining whether there are any secondary destinations nodes needing to receive program code from that destination node;

if secondary destination nodes exist, transmitting a command packet to said secondary destination nodes;

in response to said command packet, beginning independent program code downloading at each of said secondary destination nodes, by said secondary destination nodes sending individual block requests to said destination node for said destination node to transmit blocks containing said program code; and

repeating the above method steps iteratively at each secondary destination node until there are no nodes remaining in the network that need to receive program code.

6. The method according to claim 5 wherein said command packets are transmitted to said secondary destination nodes by said network controller.

7. The method according to claim 5 wherein said command packets are transmitted to said secondary destination nodes by said destination node.

8. In a wireless mesh packet communications network, said network comprising a plurality of intelligent nodes interconnected by wireless links, method for transmitting the large data file to the plurality of intelligent nodes in a network, comprising the steps of:

acquiring the large data file at a source node dividing the large data file into blocks of a size easily transmitted on the network;

transmitting a command packet to a plurality of destination nodes said command packet directing said destination nodes to request program code downloading;

in response to said command packet, beginning data file transfer from said source node to said destination nodes by said destination nodes each sending individual transmit requests to said source node for said source node to transmit said large data file.

9. The method according to claim 8 wherein said command packet includes an address for a source node and a block list identifying blocks that comprise the data file.

10. The method according to claim 9 wherein said command packet includes a list of secondary destination nodes.

First Hit Fwd Refs

☐ Generate Collection

L8: Entry 8 of 17 File: USPT Feb 13, 2001

DOCUMENT-IDENTIFIER: US 6188675 B1
TITLE: System and method for self-identifying and configuring the nodes of a network

Application Filing Date (1):
19960823

Brief Summary Text (2):
The present invention relates in general to communication networks, and more particularly, to a system and method for self-identifying and configuring the interconnected nodes of a network having an unknown or partially unknown topology. The plurality of interconnected nodes of the network includes multiple switch nodes connected together by links.

CLAIMS:

21. The self-identifying network of claim 18, wherein each node of said plurality of interconnected nodes includes a node address register and a managing node address register, and wherein said self-identifying network further comprises means for configuring said multiple switch nodes and multiple end nodes from said at least one managing node by writing a specific node address into said address node register and a managing node address into said managing node address register of each switch node and end node having unconfigured address node and managing node address registers.